



**Computación en la Nube**

# **Informe Práctica 2**



**Universidad  
de La Laguna**

## Práctica 2

1. Despliega en el laas de la ULL un cluster de 8 nodos con un core cada uno con la posibilidad de establecer comunicaciones entre ellos (port tcpip).

Para esta apartado se han creado 8 máquinas en el “laas” y se ha seguido el siguiente tutorial: <https://mpitutorial.com/tutorials/running-an-mpi-cluster-within-a-lan/>. De igual forma, se ha comprobado que todo funciona correctamente usando el código del “Token Ring” generado en la práctica anterior. Con esto he obtenido la siguiente salida:

```
[ansible@Omar-Master nfs]$ mpirun -np 8 --hostfile host ./a.out
Mensaje Enviado Desde el 0 a 1
Mensaje Recibido por 1
Mensaje Recibido por 2
Mensaje Enviado por 1 a 2
Mensaje Enviado por 2 a 3
Mensaje Recibido por 4
Mensaje Recibido por 3
Mensaje Enviado por 3 a 4
Mensaje Enviado por 4 a 5
Mensaje Recibido por 5
Mensaje Recibido por 6
Mensaje Enviado por 5 a 6
Mensaje Enviado por 6 a 7
Mensaje Recibido por 7
Mensaje Enviado por 7 a 0
Mensaje Recibido Desde el 0
```

2. El objetivo de este ejercicio es comprobar experimentalmente el costo de las comunicaciones entre pares de procesadores mediante ping-pong. Se trata además de comparar el coste de las comunicaciones con el coste de hacer una operación de tipo aritmético en el cluste creado con anterioridad.

- a) Analiza cuál debería ser la salida de los programas prod.c y ptop.c. Compila bajo MPI los programas prod.c y ptop.c. Debes ejecutar el programa prod.c con un único procesador y el programa ptop.c únicamente con dos procesadores.
- b) Representa gráficamente la salida que has obtenido con el programa ptop. Utiliza un paquete estadístico o una hoja de cálculo para realizar la regresión lineal de los datos obtenidos con el programa ptop. Representa gráficamente el ajuste y los datos obtenidos experimentalmente.
- c) Compara lo obtenido con lo obtenido en la práctica anterior.

a) Analiza cuál debería ser la salida de los programas prod.c y ptop.c. Compila bajo MPI los programas prod.c y ptop.c. Debes ejecutar el programa prod.c con un único procesador y el programa ptop.c únicamente con dos procesadores.

El fichero “prod.c” nos proporciona el tiempo que ha tardado por realizar cada operación. Su salida es la siguiente:

```
[ansible@Omar-Master nfs]$ mpirun -np 1 --hostfile host ./prod
Process 0 of 1 on Omar-Master
wall clock time = 3.854677, Prod time: 0.0000000038546774, x = 1000000000.000000
```

El fichero “ptop.c” nos proporciona el tiempo que ha tardado en realizar una comunicación con otro proceso. Su salida es la siguiente:

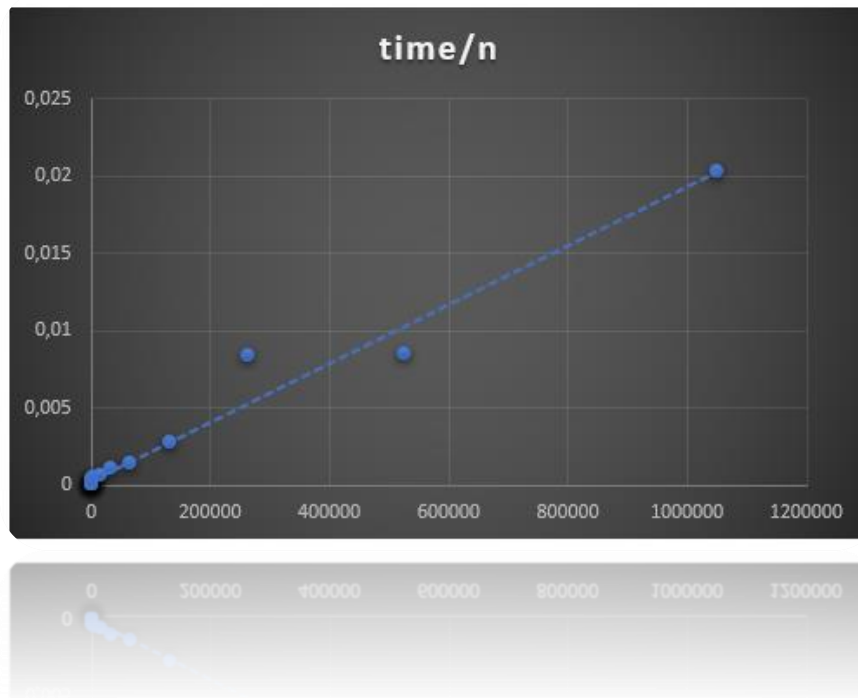
```
[ansible@Omar-Master nfs]$ mpirun -np 2 --hostfile host ./ptop
Procesador: Omar-Master
Procesador: Omar-Nodo1
Kind          n      time (sec)      MB / sec
Send/Recv     1       0.000156        0.051139
Send/Recv     2       0.000078        0.205628
Send/Recv     4       0.000066        0.487652
Send/Recv     8       0.000066        0.968557
Send/Recv    16       0.000073        1.756880
Send/Recv    32       0.000406        0.629824
Send/Recv    64       0.000066        7.707172
Send/Recv   128       0.000061       16.833562
Send/Recv   256       0.000067       30.498126
Send/Recv   512       0.000075       54.542428
Send/Recv  1024       0.000131       62.474976
Send/Recv  2048       0.000239       68.440191
Send/Recv  4096       0.000468       70.065676
Send/Recv  8192       0.000480      136.655042
Send/Recv 16384       0.000581      225.450567
Send/Recv 32768       0.001052      249.088760
Send/Recv 65536       0.001416      370.305129
Send/Recv 131072      0.002740      382.749663
Send/Recv 262144      0.008330      251.760470
Send/Recv 524288      0.008534      491.476218
Send/Recv 1048576     0.020268      413.891517
```



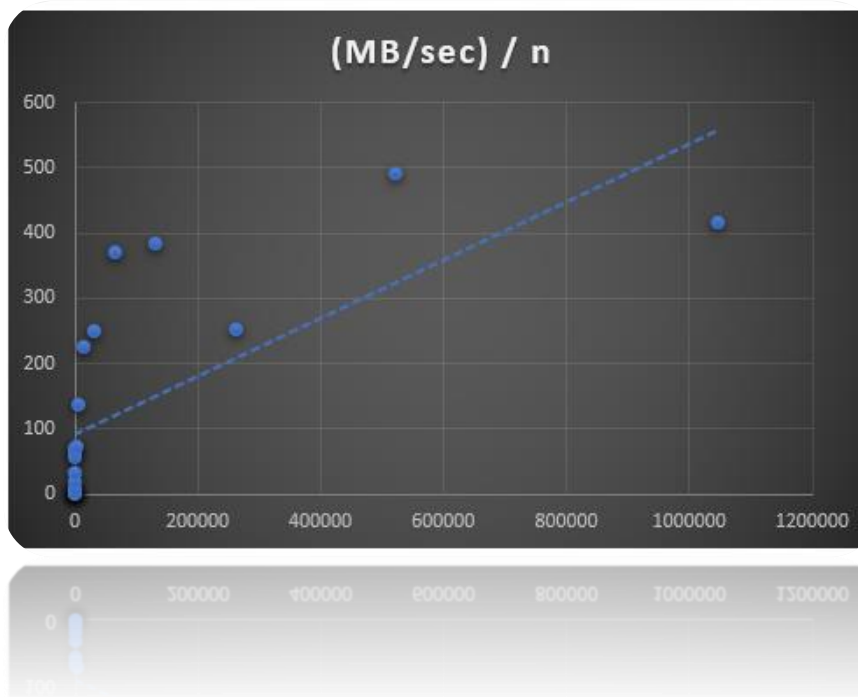
b) Representa gráficamente la salida que has obtenido con el programa ptop. Utiliza un paquete estadístico o una hoja de cálculo para realizar la regresión lineal de los datos obtenidos con el programa ptop. Representa gráficamente el ajuste y los datos obtenidos experimentalmente.

Las gráficas de regresión lineal generadas mediante Excel han sido las siguientes:

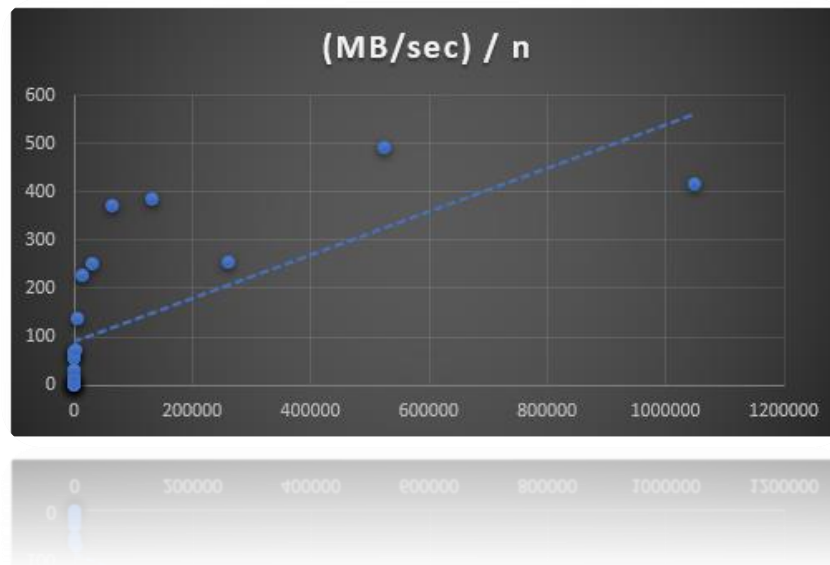
**Tiempo/ Número de comunicaciones**



**(MB/sec)/ Número de comunicaciones**



(MB/sec)/ tiempo de comunicación



### c) Compara lo obtenido con lo obtenido en la práctica anterior.

Como podemos ver en las tablas que vienen a continuación, el resultado es mucho mejor si se utiliza un procesador con varios cores en vez de utilizar la estructura generada para esta práctica. Esto se debe principalmente a que los costes de conexión entre cores de distintas máquinas es mucho más alto que en los de una única máquina.

### Un único procesador

```
usuario@lubuntu-1604:~/Descargas$ $mpirun -np 2 ./a.out
```

Kind	n	time (sec)	MB / sec
Send/Recv	1	0.000000	25.980977
Send/Recv	2	0.000000	52.143639
Send/Recv	4	0.000000	104.530941
Send/Recv	8	0.000000	174.308738
Send/Recv	16	0.000000	352.232768
Send/Recv	32	0.000000	610.752230
Send/Recv	64	0.000000	1039.104991
Send/Recv	128	0.000001	1582.356372
Send/Recv	256	0.000001	2061.584302
Send/Recv	512	0.000002	2021.161080
Send/Recv	1024	0.000003	3272.356035
Send/Recv	2048	0.000004	4042.322161
Send/Recv	4096	0.000007	4739.274258
Send/Recv	8192	0.000011	5726.623061
Send/Recv	16384	0.000021	6108.397932
Send/Recv	32768	0.000044	6024.721248
Send/Recv	65536	0.000078	6724.841760
Send/Recv	131072	0.000164	6411.146518
Send/Recv	262144	0.000402	5217.137024
Send/Recv	524288	0.001459	2875.714923
Send/Recv	1048576	0.004257	1970.340600

## Cluster

```
ansible@Omar-Master nfs]$ mpirun -np 2 --hostfile host ./ptop
```

Procesador: Omar-Master

Procesador: Omar-Nodo1

Kind	n	time (sec)	MB / sec
Send/Recv	1	0.000156	0.051139
Send/Recv	2	0.000078	0.205628
Send/Recv	4	0.000066	0.487652
Send/Recv	8	0.000066	0.968557
Send/Recv	16	0.000073	1.756880
Send/Recv	32	0.000406	0.629824
Send/Recv	64	0.000066	7.707172
Send/Recv	128	0.000061	16.833562
Send/Recv	256	0.000067	30.498126
Send/Recv	512	0.000075	54.542428
Send/Recv	1024	0.000131	62.474976
Send/Recv	2048	0.000239	68.440191
Send/Recv	4096	0.000468	70.065676
Send/Recv	8192	0.000480	136.655042
Send/Recv	16384	0.000581	225.450567
Send/Recv	32768	0.001052	249.088760
Send/Recv	65536	0.001416	370.305129
Send/Recv	131072	0.002740	382.749663
Send/Recv	262144	0.008330	251.760470
Send/Recv	524288	0.008534	491.476218
Send/Recv	1048576	0.020268	413.891517

26uq\B6CA	1048210	0'050508	473'801211
26uq\B6CA	254588	0'008234	401'410318
26uq\B6CA	505144	0'008330	521'100410
26uq\B6CA	131013	0'005140	385'140003
26uq\B6CA	02230	0'001410	310'302130
26uq\B6CA	35108	0'001023	340'088100
26uq\B6CA	10200	0'000701	500'000203